UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/037,655	01/03/2002	Jianhui Li	42390P13146	6549

8791 7590 06/23/2005

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
CHOW, CHIH CHING	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 06/23/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/037,655	LI ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 January 2002*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-60* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-60* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 January 2002* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *03/29/02*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed on January 03, 2002.

2. The priority date considered for this application is January 03, 2002.

3. Claims 1-60 have been examined.

Drawings

4. The drawings are objected to because FIG. 7A, box 720, the OBJ(j) should be OBF(j), see description in paragraph 0052 of current application. A corrected drawing is needed. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless -
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 1-31, 33, 39, 41, 43, 49, 51, 53 are rejected under 35 U.S.C. 102(e) as being anticipated by US2004/0268094 by Abdallah et al. (hereinafter "Abdallah").

The applied reference has a common Assignee with the instant application. Based upon the earlier effective U.S. filing date of the reference, it constitutes prior art under 35 U.S.C. 102(e). This rejection under 35 U.S.C. 102(e) might be overcome either by a showing under 37 CFR 1.132 that any invention disclosed but

not claimed in the reference was derived from the inventor of this application and is thus not the invention "by another," or by an appropriate showing under 37 CFR 1.131.

CLAIM	Abdallah
1. A method comprising: determining a register format of a source register operated on by a source instruction in a source block of code, the register format including an input instruction format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and being used as an input of the source instruction, the input instruction format containing format of the source register expected by the source instruction, the output block format containing format of the source register after the source block of code is executed; and detecting an instruction format inconsistency between the source register and a target register of a target architecture during a translation phase of a binary translation that translates the source block of code into a target block of code running in the target architecture.	Abdallah teaches converting instruction format from one register to a different register format. See Abdallah's paragraph 49, "**data in one format in one architectural register is converted to another format and placed in another architectural register.**" Also in Abdallah's Abstract, "Numbers are stored in the **integer format in a register of a first set of architectural registers** in a packed format. At least one of the numbers in the integer format is **converted to at least one number in the floating point format**. The numbers in the floating point format are placed in a register of a **second set of architectural registers** in a packed format" (*in Abdallah's disclosure, it implies that 'determining a register format of a source register' first followed by a 'detecting an instruction format inconsistency between the source register and a target register' step, otherwise it can't convert from the original one to another format).*
2. The method of claim 1 wherein detecting the instruction format inconsistency comprises:	For the feature of claim 1 see claim 1 rejection. If the output block format is floating point format, and the input

comparing the output block format to the input instruction format if the output block format asserts an access status of the source register.	instruction format is integer or vice versa, the comparison needs to be done first before the conversion is done. See claim 1 rejection.
3. The method of claim 2 further comprising: emitting a conversion code to convert the source register from the output block format to the input instruction format into the target block of code during the translation phase if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.	For the feature of claim 2 see claim 2 rejection. See Abdallah FIG. 10, the conversion is done assuming the input instruction is in Floating Point format, and the output block is in Integer format. Also see Abdallah's paragraph 0049, "the conversion results would be immediately required from memory, necessitating a memory access operation that would place the results back in a register".
4. The method of claim 1 further comprising; updating an input block format and the output block format, the input block format containing format of the source register expected by the source block of code before execution.	Same as claim 1 rejection.
5. The method of claim 4 wherein updating comprises: setting the input block format and the output block format to the input instruction format if the output block format does not assert an access status of the source register.	For the feature of claim 4 see claim 4 rejection. For the rest of claim 5 feature see claim 3 rejection.
6. The method of claim 4 wherein updating comprises: setting the output block format to the	For the feature of claim 4 see claim 4 rejection. For the rest of claim 6 feature see claim 3 rejection

input instruction format if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.	
7. The method of claim 4 wherein updating comprises: setting the output block format to the output instruction format for the source register being used as output of the source instruction.	For the feature of claim 4 see claim 4 rejection. For the rest of claim 7 feature see claim 3 rejection
8. The method of claim 1 further comprising: a. emitting a target instruction sequence corresponding to the source instruction into the target block of code; b. emitting a block inconsistency check code into prefix of the target block of code; and c. emitting a format update code to update a format register associated register format into the suffix of the target block of code.	For the feature of claim 1 see claim 1 rejection. For item a, see claim 3 rejection, for item b, see Abdallah's paragraph 107, "this embodiment comprises four or more bytes. In addition to the control signal format of FIG. 6a, the control signal format of FIG. 6b includes **a prefix** 613." For item c, also see FIG 6a and 6b (*DEST is the target code, suffix is the other end of the prefix*).
9. The method of claim 1 wherein determining the register format comprises: determining one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.	Same as claim 1 rejection.
10. The method of claim 8 wherein	For the feature of claim 4 see claim 4

emitting the block inconsistency check code comprises:

emitting the block inconsistency check code to be executed during an execution phase following the translation phase.

rejection. Abdallah's disclosure also teaches the block inconsistency check code to be executed during an execution, see paragraph 0012, "A prior art method of dealing with this problem duplicates the floating point execution resources of the processor. This duplication of resources allows for two floating point pipelines executing at the same time wherein the floating point data of each branch of the pipeline can be sequentially converted to integer format at the same time."

11. A computer program product comprising:

a machine useable medium having program code embedded therein, the program code comprising:

computer readable program code to determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input instruction format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and being used as an input of the source instruction, the input instruction format containing format of the source register expected by the source instruction, the output block format containing format of the source register after the source block of code is executed; and

Abdallah's disclosure is implemented in a machine useable medium, and a computer readable program code. See claim 1 rejection.

computer readable program code to detect an instruction format inconsistency between the source register and a target register of a target architecture during a translation phase of a binary translation that translates the source block of code into a target block of code running in the target architecture.

12. The computer program product of claim 11 wherein the computer readable program code to detect the instruction format inconsistency comprises:
computer readable program code to compare the output block format to the input instruction format if the output block format asserts an access status of the source register.

For the feature of claim 11 see claim 11 rejection. For the rest of claim 12 feature see claim 1 and 2 rejections.

13. The computer program product of claim 12 further comprising:
computer readable program code to emit a conversion code to convert the source register from the output block format to the input instruction format into the target block of code during the translation phase if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

For the feature of claim 12 see claim 12 rejection. For the rest of claim 13 feature see claim 3 rejection.

14. The computer program product of claim 11 further comprising:
computer readable program code to

For the feature of claim 11 see claim 11 rejection. For the rest of claim 14 feature see claim 4 rejection.

update an input block format and the output block format, the input block format containing format of the source register expected by the source block of code before execution.

15. The computer program product of claim 14 wherein the computer readable program code to update comprises:
computer readable program code to set the input block format and the output block format to the input instruction format if the output block format does not assert an access status of the source register.

For the feature of claim 14 see claim 14 rejection. For the rest of claim 15 feature see claim 5 rejection.

16. The computer program product of claim 14 wherein the computer readable program code to update comprises:
computer readable program code to set the output block format to the input instruction format if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

For the feature of claim 14 see claim 14 rejection. For the rest of claim 16 feature see claim 6 rejection.

17. The computer program product of claim 14 wherein the computer readable program code to update comprises:
computer readable program code to set the output block format to the output instruction format for the source register being used as output of the source instruction.

For the feature of claim 14 see claim 14 rejection. For the rest of claim 17 feature see claim 7 rejection.

18. The computer program product of claim 11 further comprising: computer readable program code to emit a target instruction sequence corresponding to the source instruction into the target block of code; computer readable program code to emit a block inconsistency check code into prefix of the target block of code; and computer readable program code to emit a format update code to update a format register associated register format into the suffix of the target block of code.	For the feature of claim 11 see claim 11 rejection. For the rest of claim 18 feature see claim 8 rejection.
19. The computer program product of claim 11 wherein the computer readable program code to determine the register format comprises: computer readable program code to determine one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.	For the feature of claim 11 see claim 11 rejection. For the rest of claim 19 feature see claim 9 rejection.
20. The computer program product of claim 18 wherein the computer readable program code to emit the block inconsistency check code comprises: computer readable program code to emit the block inconsistency check code to be executed during an execution phase following the translation phase.	For the feature of claim 18 see claim 18 rejection. For the rest of claim 20 feature see claim 10 rejection.
21. A system comprising:	Abdallah's disclosure also is a system

a processor; and

a memory coupled to the processor to store program code, the program code, when executed, causing the processor to:

determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input instruction format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and being used as an input of the source instruction, the input instruction format containing format of the source register expected by the source instruction, the output block format containing format of the source register after the source block of code is executed; and

detect an instruction format inconsistency between the source register and a target register of a target architecture during a translation phase of a binary translation that translates the source block of code into a target block of code running in the target architecture.

comprising a processor, a memory coupled to the process to store program code. See claim 1 rejection.

22. The system of claim 21 wherein the program code causing the processor to detect the instruction format inconsistency causes the processor to:

compare the output block format to the input instruction format if the

For the feature of claim 21 see claim 21 rejection. For the rest of claim 22 feature see claim 2 rejection.

output block format asserts an access status of the source register.

23. The system of claim 22 wherein the program code further causing the processor to:
emit a conversion code to convert the source register from the output block format to the input instruction format into the target block of code during the translation phase if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

For the feature of claim 22 see claim 22 rejection. For the rest of claim 23 feature see claim 3 rejection.

24. The system of claim 21 wherein the program code further causing the processor to:
update an input block format and the output block format, the input block format containing format of the source register expected by the source block of code before execution.

For the feature of claim 21 see claim 21 rejection. For the rest of claim 24 feature see claim 4 rejection.

25. The system of claim 24 wherein the program code causing the processor to update causes the processor to:
set the input block format and the output block format to the input instruction format if the output block format does not assert an access status of the source register.

For the feature of claim 24 see claim 24 rejection. For the rest of claim 25 feature see claim 5 rejection.

26. The system of claim 24 wherein the program code causing the processor to

For the feature of claim 24 see claim 24 rejection. For the rest of claim 26

update causes the processor to:
set the output block format to the input instruction format if the output block format is different from the input instruction format and the output block format asserts an access status of the source register.

feature see claim 6 rejection.

27. The system of claim 24 wherein the program code causing the processor to update causes the processor to:
set the output block format to the output instruction format for the source register being used as output of the source instruction.

For the feature of claim 24 see claim 24 rejection. For the rest of claim 27 feature see claim 7 rejection.

28. The system of claim 21 the program code further causing the processor to:
emit a target instruction sequence corresponding to the source instruction into the target block of code;
emit a block inconsistency check code into prefix of the target block of code; and
emit a format update code to update a format register associated register format into the suffix of the target block of code.

For the feature of claim 21 see claim 21 rejection. For the rest of claim 28 feature see claim 8 rejection.

29. The system of claim 21 wherein the program code causing the processor to determine the register format causes the processor to:
determine one of an access status, a packed single precision format, a packed double precision format, and a packed

For the feature of claim 21 see claim 21 rejection. For the rest of claim 29 feature see claim 9 rejection.

integer format.

30. The system of claim 28 wherein the program code causing the processor to emit the block inconsistency check code causes the processor to: emit the block inconsistency check code to be executed during an execution phase following the translation phase.	For the feature of claim 28 see claim 28 rejection. For the rest of claim 30 feature see claim 10 rejection.
31. A method comprising: determining a register format of a source register operated on by a source instruction in a source block of code, the register format including an input block format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and a format register associated with the register format, the input block format containing format of the source register expected by the source block of code, the output block format containing format of the source register after the source block of code is executed; and detecting a block format inconsistency between the source register and a target register of a target architecture during an execution phase of a binary translation that translates the source block of code into a target block of code running in the target architecture.	Same as claim 1 rejection.
33. The method of claim 31 further	For the feature of claim 31 see claim 31

comprising: updating the format register upon exit of the target block of code.	rejection. From Claim 31, a block means code between an entry and an exit point, the updating occurs after detecting the block and the exit of the target block of code.
39. The method of claim 31 wherein determining the register format comprises: determining one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.	For the feature of claim 31 see claim 31 rejection. For the rest of claim 39 feature see claim 9 rejection.
41. A computer program product comprising: a machine useable medium having program code embedded therein, the program code comprising: computer readable program code to determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input block format and an output block format of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and a format register associated with the register format, the input block format containing format of the source register expected by the source block of code, the output block format containing format of the source register after the source block of code is executed; and	Same as claim 1 rejection.

computer readable program code to detect a block format inconsistency between the source register and a target register of a target architecture during an execution phase of a binary translation that translates the source block of code into a target block of code running in the target architecture.

43. The computer program product of claim 41 further comprising:
computer readable program code to update the format register upon exit of the target block of code.

For the feature of claim 31 see claim 41 rejection. For the rest of claim 43 feature see claim 33 rejection.

49. The computer program product of claim 41 wherein the computer readable program code to determine the register format comprises:
computer readable program code to determine one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.

For the feature of claim 41 see claim 41 rejection. For the rest of claim 49 feature see claim 9 rejection.

51. A system comprising:
a processor; and
a memory coupled to the processor to store program code, the program code, when executed, causing the processor to:
determine a register format of a source register operated on by a source instruction in a source block of code, the register format including an input block format and an output block format

Same as claim 1 rejection.

of the source block of code, the source block of code running in a source architecture, the source register having multiple formats and a format register associated with the register format, the input block format containing format of the source register expected by the source block of code, the output block format containing format of the source register after the source block of code is executed; and

detect a block format inconsistency between the source register and a target register of a target architecture during an execution phase of a binary translation that translates the source block of code into a target block of code running in the target architecture.

53. The system of claim 51 the program code further causing the processor to:

update the format register upon exit of the target block of code.

For the feature of claim 51 see claim 51 rejection. For the rest of claim 53 feature see claim 33 rejection.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 32, 34-38, 40, 42, 44-48, 50, 52, 54-60 are rejected under 35 U.S.C. 103(a) as being unpatentable over US2004/0268094 by Abdallah et al. (hereinafter "Abdallah"), in view of U.S. Patent No. 6,789,181 by John Yates et al. (hereinafter "Yates").

CLAIM	Abdallah / Yates
32. The method of claim 31 wherein detecting the block format inconsistency comprises: masking the format register with an input block format mask; and comparing the masked format register with the input block format.	For the feature of claim 31 see claim 31 rejection. Abdallah teaches all aspects of claim 32, but he does not mention 'masking the format register' specifically, however, Yates teaches it in an analogous prior art. In Yates' column 7, lines 29-31, "The recorded profile information may record a data-dependent change to a full/empty **mask** for registers of the computer." And column 11, lines 66-67 into column 12 line 5, "The pipeline control circuitry is designed to control processing of instructions by the instruction pipeline circuitry as part of the basic instruction processing cycle of the microprocessor, depending, at least in part, on the value of the entry corresponding to the address range in which lies an instruction processed by the instruction pipeline circuitry, and the current value of the **mask**." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement Abdallah's disclosure of the regiser format converter by masking the registers taught by Yates, for the purpose of ensuring the instruction is

	within address range (See Yates' column 11 line 60 into column 12 line 24).
34. The method of claim 33 wherein updating the format register comprises: generating a first comparison result between the format register and the output block format; masking the first comparison result by an output block format mask; and generating a second comparison result between the format register and the masked first comparison result, the second comparison result corresponding to the updated format register.	For the feature of claim 33 see claim 33 rejection. For the rest of claim 34 feature see claim 1 and 32 rejections.
35. The method of claim 32 further comprising: executing a self-correcting code if the masked input block format is different than the input block format.	For the feature of claim 32 see claim 32 rejection. For the rest of claim 35 feature see paragraphs 0212-0214 for self-correcting code process.
36. The method of claim 35 wherein executing comprises: asserting a correction condition based on the format register and the input block format.	For the feature of claim 35 see claim 35 rejection. For the rest of claim 36 feature see claim 35 rejection.
37. The method of claim 36 wherein asserting comprises: comparing the format register to the input block format if the input block format asserts an access status of the source register.	For the feature of claim 36 see claim 36 rejection. For the rest of claim 37 feature see claim 2 rejection.
38. The method of claim 36 further	For the feature of claim 36 see claim 36

comprising: converting the source register from format contained in the format register to format contained in the input block format; and setting the format register to the input block format.	rejection. For the rest of claim 38 feature see claim 3 and 5 rejections.
40. The method of claim 38 wherein detecting the block format inconsistency comprises: detecting the block format inconsistency during the execution phase that follows a translation phase in the binary translation.	For the feature of claim 38 see claim 38 rejection. For the rest of claim 40 feature see claim 10 rejection.
42. The computer program product of claim 41 wherein the computer readable program code to detect the block format inconsistency comprises: computer readable program code to mask the format register with an input block format mask; and computer readable program code to compare the masked format register with the input block format.	For the feature of claim 41 see claim 41 rejection. For the rest of claim 42 feature see claim 32 rejection.
44. The computer program product of claim 43 wherein the computer readable program code to update the format register comprises: computer readable program code to generate a first comparison result between the format register and the output block format; computer readable program code to	For the feature of claim 43 see claim 43 rejection. For the rest of claim 44 feature see claim 34 rejection.

mask the first comparison result by an output block format mask; and
computer readable program code to generate a second comparison result between the format register and the masked first comparison result, the second comparison result corresponding to the updated format register.

45. The computer program product of claim 42 further comprising:
computer readable program code to execute a self-correcting code if the masked input block format is different than the input block format.

For the feature of claim 42 see claim 42 rejection. For the rest of claim 45 feature see claim 35 rejection.

46. The computer program product of claim 45 wherein the computer readable program code to execute comprises:
computer readable program code to assert a correction condition based on the format register and the input block format.

For the feature of claim 42 see claim 42 rejection. For the rest of claim 45 feature see claim 36 rejection.

47. The computer program product of claim 46 wherein the computer readable program code to assert comprises:
computer readable program code to compare the format register to the input block format if the input block format asserts an access status of the source register.

For the feature of claim 46 see claim 46 rejection. For the rest of claim 47 feature see claim 2 rejection.

48. The computer program product of claim 46 further comprising:
computer readable program code to

For the feature of claim 46 see claim 46 rejection. For the rest of claim 48 feature see claim 38 rejection.

convert the source register from format contained in the format register to format contained in the input block format; and

computer readable program code to set the format register to the input block format.

50. The computer program product of claim 48 wherein the computer readable program code to detect the block format inconsistency comprises:

computer readable program code to detect the block format inconsistency during the execution phase that follows a translation phase in the binary translation.

For the feature of claim 48 see claim 48 rejection. For the rest of claim 50 feature see claim 2 rejection.

52. The system of claim 51 wherein the program code causing the processor to detect the block format inconsistency causes the processor to:

mask the format register with an input block format mask; and

compare the masked format register with the input block format.

For the feature of claim 51 see claim 51 rejection. For the rest of claim 52 feature see claim 32 rejection.

54. The system of claim 53 wherein the program code causing the processor to update the format register causes the processor to:

generate a first comparison result between the format register and the output block format;

mask the first comparison result by an output block format mask; and

For the feature of claim 53 see claim 53 rejection. For the rest of claim 54 feature see claim 34 rejection.

generate a second comparison result between the format register and the masked first comparison result, the second comparison result corresponding to the updated format register.

55. The system of claim 52 wherein the program code further causing the processor to:
execute a self-correcting code if the masked input block format is different than the input block format.

For the feature of claim 52 see claim 52 rejection. For the rest of claim 55 feature see claim 35 rejection.

56. The system of claim 55 the program code causing the processor to execute causes the processor to:
assert a correction condition based on the format register and the input block format.

For the feature of claim 55 see claim 55 rejection. For the rest of claim 56 feature see claim 36 rejection.

57. The system of claim 56 the program code causing the processor to assert causes the processor to:
compare the format register to the input block format if the input block format asserts an access status of the source register.

For the feature of claim 56 see claim 56 rejection. For the rest of claim 57 feature see claim 37 rejection.

58. The system of claim 56 wherein the program code further causing the processor to:
convert the source register from format contained in the format register to format contained in the input block format; and
set the format register to the input

For the feature of claim 56 see claim 56 rejection. For the rest of claim 58 feature see claim 38 rejection.

block format.

59. The system of claim 51 wherein the program code causing the processor to determine the register format causes the processor to: determine one of an access status, a packed single precision format, a packed double precision format, and a packed integer format.	For the feature of claim 51 see claim 51 rejection. For the rest of claim 59 feature see claim 9 rejection.
60. The system of claim 58 wherein the program code causing the processor to detect the block format inconsistency causes the processor to: detect the block format inconsistency during the execution phase that follows a translation phase in the binary translation.	For the feature of claim 58 see claim 58 rejection. For the rest of claim 60 feature see claim 40 rejection.

Conclusion

The following summarizes the status of the claims:

35 USC § 102 rejection: Claims 1-31, 33, 39, 41, 43, 49, 51, 53

35 USC § 103 rejection: Claims 32, 34-38, 40, 42, 44-48, 50, 52, 54-60

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 571-272-3693. The examiner can normally be reached on 7:30am - 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned

is 703-872-9306. Any inquiry of a general nature of relating to the status of this application should be directed to the **TC2100 Group receptionist: 571-272-2100**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow

Examiner

Art Unit 2192

June 17, 2005

CC

ANTONY NGUYEN-BA
PRIMARY EXAMINER